AUSTRALIAN OS9 NEWSLETTER

Volume 7 | September 1993 | Number 8

EDITOR:	Gordon Bentzen	(07) 344-3881
SUB-EDITOR:	Bob Devries	(07) 278-7209
TREASURER:	Jean-Pierre Jacquet	(07) 372-4675
	Fax Messages	(07) 372-8325
LIBRARIAN:	Rod Holden	(07) 200-9870
CONSULTANT:	Don Berrie	(079) 75-3537
SUPPORT:	Brisbane OS9 Users Group	

OS9 OS9

OS9 OS9

OS9 OS9

Editorial Material:
Gordon Bentzen
8 Odin Street
SUNNYBANK QLD 4109

Library Requests:
Rod Holden
53 Haig Road
LOGANLEA QLD 4131

AUSTRALIAN OS9 NEWSLETTER
Newsletter of the National OS9 User Group
Volume 7 Number 8

EDITOR : Gordon Bentzen
SUBEDITOR : Bob Devries
TREASURER : Jean-Pierre Jacquet
LIBRARIAN : Rod Holden
SUPPORT : Brisbane OS9 Level 2 Users Group.

HERE WE GO AGAIN!

Our August editorial advised that we would require a minimum of twenty members for the coming year, plus at least three members to contribute articles on a regular basis.

Well here is a progress report on our membership renewal. We have already received twenty-one renewals at the time of writing, so this means that we have achieved the subscription renewals needed for us to meet the criteria of a minimum twenty members.

A couple of members have also indicated that they would be able to contribute articles to the newsletter, and although this is not the strong overall commitment we had hoped for, this does mean that we WILL do it all again.

Please do send material for inclusion to the editor if you can, as this will make each edition more interesting for everyone.

THE LATEST FROM THE U.S. USER'S GROUP

Since our last edition we have received a "Special Edition!! MOTD" (Message Of The Day). This four page newsletter was rushed out to advise of recent announcements concerning the U.S. OS-9 Users Group.

Firstly:- It is confirmed that Jim DeStafeno, president since April 1, 1993, has indeed resigned as president of the group. This of course is his prerogative, however without the approval of the board of directors, Mr DeStafeno suddenly cancelled all memberships and dispersed the Users Group funds.

Secondly:- Steps have been taken to revitalise the OS-9 Users Group by the Board of Directors, Boisy Pitre, Carl Kreider and George Dorner. Support has been pledged by the Chicago Area OS-9 Users Group and members, Carl Boll, Allen Huffman and Joel Hegberg. A new mail address is also announced.

The OS-9 Users Group, Inc.
6158 W 63rd Street
Suit 109
Chicago Ill 60638.

EUROS9 USERGROUP

It seems that Peter Tutelaers of the Euros9 Usergroup has also been able to find the time to continue with the OS-9 "Co-operation" which he has worked so hard at. Peter also still has a number of copies of the book "The OS-9 GURU by Paul S. Dayan" which were unsold at the Chicago CoCoFest in May. This is a very detailed and technical publication dealing with OSK (68000 OS-9) and was selling at US$40.

Please contact me if you are interested in a copy as I am sure that we could arrange supplies from Peter.

THE "ROCKET"

For those not familiar with this topic, the "Rocket" proposed by Burke & Burke is a circuit board with 68306 CPU, 2 SIMM sockets, a bus synchroniser, 32K byte on-board ROM, 6X09 CPU socket, mode switch, and a limited (no bus) I/O expansion connector. The board plugs into the CoCo 3 CPU socket and has a "mode" switch to select the 6x09 or the 68306. It would come with OS-9 68000.

Chris Burke was looking for firm orders of 100 units before going into production.

Well the progress is slow, less than 100 orders received, and many other cost increases and difficulties have the whole project in considerable doubt. Chris Burke has called for suggestions. It seems that "Rocket" may not get past the prototype stage.

That's about it for this month. Our sincere thanks to those who have already renewed their support of the Australian OS-9 Usergroup and also to those who are about to. Cheers, Gordon.

Please Note:- This is the last issue to be sent to members who have not yet renewed their subscription. Please check the back page and advise us of any errors or omissions.

The National OS9 Usergroup
(07)-200-9870
300/1200/2400 baud.
20:00 to 22:30 HRS.(AEST)
(8N1)

Co-ordinator: Bob Devries (07)-278-7209
Sysop: Rod Holden

This is (RiBBS).... A Tandy Coco Based BBS program.
This BBS is accessable to Usergroup Members ONLY!
Feel free to look around , and test out the options.

OS9 for Ever !!!!

Hi, this is your Sysop once again letting you know what type of software is available. Here is a utility program which you will find very useful for duplicating files.

DupFile - File duplication utility
By: Brian C. White (BRIANWHITE)

DupFile will take any file or directory and create a duplicate of it on the same disk. Basically, all that is done is that a new name is created that points to the same file and the file's link count is then incremented.

The creators of OS-9 made a provision for this. The "del" command works much like the "unlink" command. First, it decreses the link count of the file by one. If the link count is then zero, it re-allocates all of the file's sectors. Thus, if you delete a file that you have duplicated, the name you deleted will be erased from the directory, but another name, elsewhere on the disk will still point to the file and the file will NOT be erased from the disk. A maximum of 255 names can point to one file.

The syntax for DupFile is:

DupFile <Device> <Original> <Duplicate>

<Device> is the device on which the file is located. (ie. /d0 /d1 /h0 /r0 etc...)

<Original> is the path to the original file from the root directory of the specified device.

<Duplicate> is the path to the new duplicated file from the root directory of the specified device.

So, to duplicate the file "/d0/MYFILES/neat_text"
to "/d0/YOURFILES/my_neat_text"
you would use a command line like:

DupFile /d0 MYFILES/neat_text YOURFILES/my_neat_text

Directories can be duplicated in the exact same way. Just end the pathlist with a directory name instead of a file name. Both directories will then contain the same list of files at all times. Remember that all duplicated directories will have the same parent directory! You may wish to take a disk editor and manually change the first byte in the double period (..) to a $FF or some other unaccessable character. [NOT a good idea, some programmes depend on the . and .. as filenames. ED] Do not delete it because if another file were written over it, dir would not see it due to the fact that it automatically skips the first two directory entries (period and double period). Duplicated directories work best if they are on the same level.

NOTE: Do NOT duplicate a directory so as to cause it to become its own subdirectory!

Deleting a duplicated directory can be tricky. The "DelDir" command was not programmed so as to handle duplicate files. If you use "DelDir", be sure to track down all duplicates of it and "Del" (not "DelDir) them as well.

To delete just one copy of a duplicated directory, use a disk editor, such as "dEd" and skip to the sector number given in the last three bytes of the directory entry (each entry is 32 or $20 bytes long). Now decriment byte $08 of the sector (the link count of the file) by one. Note that if this byte is already a $01, use "DelDir" instead because the directory file has no duplicates on the disk. Then, go back to the sector with the directory entry you wish to delete and change the first byte of the name to a $00, thus deleting the file.

If someone patches "DelDir" to accomodate duplicated files, be sure to let me know so I can download a copy of the patch.

One use of this program could be for organization. For example, if you use a hard disk and C, you have a directory called "LIB". Suppose some other compiler comes out that uses a directory called "LIBS". All you would have to do is:

DupFile /h0 LIB LIBS

and both directories would be the same. All of your library files could be put in the same place, but they could all be seen through either "LIB" or "LIBS"! Or, maybe you use the "c.asm" assembler to do your ML work, but you have it copied to a file called "asm" for simplicity. Instead, just do:

DupFile /h0 CMDS/c.asm CMDS/asm

and you will get the same results with no needless loss of disk space.

One other important note: Duplicated files take up only 32 bytes of disk space! Chances are that if you do a "Free" before and after the duplication process, both will return the same number of free sectors.

I have now bought a 40 track 5 1/4 drive to make life easier for those members who have only 40 track drives. I am still working on this bible scanner, so please be patient as I am only a beginner at programming. See you in the bit stream, Happy CoCoing.

Sysop
Rod Holden

PLEASE NOTE: BBS HOURS 8.00 PM to 10:30 PM
DO NOT call outside these hours
except by prior appointment

NEW LIBRARY FILES

Our PD library has recently received about 2.5MB of new files. A large number (120+, 1.82MB) of these are UltimUse scores, for those of you who own MIDI keyboards, but a number of them are also of general interest. These include:

```
about_help.txt   ---\
basichelp.ar         \
clibhelp.ar           \
devpakhelp.ar          \
filespecs_help.ar       \ new help utility
help.ar                 /
helpsrc.ar             /
installing_help.txt   /
mvhelp.ar            /
windows_help.ar  ---/
termcaplvl2.ar   TERMCAP library function
theo.ar          THESAURUS programme
kermit.ar        new KERMIT (V21.5)
inventory_plus21.ar INVENTORY programme
gshell32.ar         \
gshell_125.ar        / patches to GSHELL
porder.ar        PURCHASE order programme
yaipl_4.ar       INVENTORY programme
lzh10.ar         LZH archiver
unixinclude.lzh  \
unixlib.lzh      / UNIX functions for C
unixutils.ar     UNIX like utilities
ccunzip_1_02.ar  UNZIP for OS9
unzip45b.lzh      ""       ""
sermidi.ar       programme to send MIDI
ubox3.ar         UltiBox player
```

These files will appear in the PD library, on the next available disk (12?...13) and also on the BBS very soon.

Removing TAB characters
by Bob Devries

It's funny, you know. I used to wonder where all those neat utility programmes came from, and why people wrote them. Well, I found out! I recently enrolled in an advanced C programming course at college, and as an assignment I was asked to design a programme to remove TAB characters from a file, and replace them with a variable number of spaces.

It only took a couple of hours to write and debug, and although I used my trusty CoCo3 with OS9 level 2, it had to work on a PC clone, and a UNIX machine, too. While I was at it, I also tested it on my Amiga.

The programme had to use the standard UNIX style command line (just like OS9), with the number of spaces to replace a TAB character, and the filename in it. The output was to go to STDOUT, and any problems, such as incorrect command line, and file problems also to be reported, in UNIX's terse style. Well, I felt that some of you could benefit from my sweat, so here's the code.

```
/* Detab.c */
/* Command line: detab -e # filename */
/* where # is the number of spaces to expand TAB chars to */

#include <stdio.h>
#include <ctype.h>

#define MAXTAB 20  /* maximum TAB expansion */
#define TAB '\t'
#define CR '\r'
#define LF '\n'

main(argc,argv)
int argc;
char *argv[];
{
    FILE *fopen(), *ifp;
```

```
    int count = 0;
    int tab, x, mod;
    char ch;

    if (argc < 4) {
         usage(argv[0]);
         exit(0);
    }

    if ((argv[1][0] != '-') && (tolower(argv[1][1])) != 'e') {
         usage(argv[0]);
         exit(0);
    }

    if (!(isdigit(argv[2][0]))) {  /* if arg is not numeric, exit */
         usage(argv[0]);
         exit(0);
    }

    tab = atoi(argv[2]);           /* get decimal value of TAB width */
    if ((tab < 1)    (tab > MAXTAB)) {  /* check if > MAXTAB */
         usage(argv[0]);
         exit(0);
    }

    if ((ifp = fopen(argv[3],"r")) == (FILE *)NULL) {
         fprintf(stderr,"Can't open file \'%s\'.\n",argv[3]);
         exit(errno);
    }

    while ((ch = fgetc(ifp)) != (char)EOF) {
         count++;
         if ((ch == CR)    (ch == LF)) count=0; /* reset counter if EOL */
         if (ch == TAB) {                       /* if TAB char, process */
                   count--;
                   if (count < tab) {
                        for (x=1; x<=tab; x++) {
                             putchar(' ');      /* insert spaces */
                             count++;
                        }
                   } else {
                        mod = count%tab;        /* get next TAB zone */
                        for (x=1; x<=(tab-mod); x++) {
                             putchar(' ');
                             count++;
                        }
                   }
         } else {
                   putchar(ch); /* output char if not TAB */
         }
    }

    fclose(ifp);
}
```

```
usage(thisprog) /* user error, tell him */
char *thisprog;
{
    fprintf(stderr,"Usage: %s -e # filename\n",thisprog);
    fprintf(stderr,"       where # = TAB width.\n");
    fprintf(stderr,"       TAB width 1...20.\n");
}

/* By Bob Devries. 06-AUG-93 */
/* EOF */
```

New OS9 help utility
by Tim Kientzle

This Help System is designed to correct several deficiencies with the Tandy system.

1) Easy to maintain. Adding a new description, or altering an old one, is as simple as putting a file into a particular directory. Since each description is stored in a single file, it is short and easy to edit. The "Help" program even has a shell escape to allow you to easily call up an editor from within "Help".

2) Structured. Tandy Help, like Unix Man, is "flat". I find it easier to navigate through a system where each description is short and easy to understand, and where you can choose exactly which information you want to see. Implementing a help system with "topics" and "subtopics" makes it easy to put rough descriptions for each topic, with additional details hidden in subtopics. With a "flat" system, you must either put all the details in one description, which makes it difficult to find the particular data you want, or else omit a lot of information. The drawback, of course, is that being able to manage a large database is only an asset if you have the space to store such a database.

So, I recommend that only people with plenty of disk space even consider this system. My package of CLIB help takes up over 250k of disk space by itself, and my entire Help directory is currently over 1.5 megabytes. But, if you have the hard disk space, I think it is well worth it to have as much information on-line as possible. I currently am working on getting the CGFX library on-line, as well as complete info on OS9 system calls and other RMA programming information.

MAKE
from a college text

Make is an automated program compilation utility. It determines (from a set of rules that you supply) which of the modules that make up the program you are working on need to be recompiled and then recompiles only those modules. It can also tell what needs to be recompiled when you change an include file. It does all this by keeping a list of DEPENDENCIES. You tell it that a particular file must be recompiled if a certain include file has been modified more recently than has the the C source file (or if the C source has been modified more recently than its object file).

Make takes as an input file a makefile - a file that describes all the modules in a large program and the relationships between these modules. Using the makefile, make determines what modules need to be recompiled at any given moment, and then compiles them (and only them).

Example. Assume you want to create an executable called 'farm'. The original source is split up into three files:- cow.c, pig.c and farm.c. All three files contain a '#include <stdio.h>'. Furthermore, cow.c and pig.c have a '#include "animals.h". Now, if you change something in cow.c, you'll have to recompile cow.c and then relink the cow.o into farm. If you change something in animals.h, you must recompile both cow.c and pig.c and relink. If you change something in stdio.h, you'll have to recompile

everything. Make describes the relationships between these files as DEPENDENCIES. The makefile is a list of these dependencies. Here's a simple makefile for the program above.

```
#
# Make farm using xlc
#

farm: cow.o pig.o farm.o
[TAB] xlc -o farm cow.o pig.o farm.o

cow.o: cow.c stdio.h animals.h
[TAB] xlc -c cow.c

pig.o: pig.c stdio.h animals.h
[TAB] xlc -c pig.c

farm.o: farm.c stdio.h
[TAB] xlc -c farm.c
```

The # is a comment line. farm dependes on cow.o, pig.o and farm.o. Hence, if any of the files cow.o, pig.o or farm.o have been changed more recently than farm, then farm has to be remade. This remake is accomplished by the line 'xlc -o farm cow.o pig.o farm.o'. Similarly, if any of the files pig.c, stdio.h or animals.h have been changed more recently than pig.o, then pig.o is remade using the command 'xlc -c pig.c'.

Make supports a number of options, some of them are described below.

Macros:- used to save typing and makes changes easier.

A macro is defined by:

```
macro_name=replacement_string
```

The macro_name is expanded by $(macro_name)

```
#
# Make farm .... example of macros
#

CC = xlc
INCLUDES = stdio.h animals.h
OBJECTS = cow.o pig.o farm.o

farm: $(OBJECTS)
[TAB] $(CC) -o farm $(OBJECTS)

cow.o: $(INCLUDES) cow.c
[TAB] $(CC) -c cow.c

pig.o: $(INCLUDES) pig.c
[TAB] $(CC) -c pig.c

farm.o: stdio.h farm.c
[TAB] $(CC) -c farm.c
```

Generic dependency: all .c files are converted to .o files with the same set of actions.

```
#
# Make farm .... example of macros
#

CC = xlc
INCLUDES = stdio.h animals.h
OBJECTS = cow.o pig.o farm.o

.c.o:
[TAB] $(CC) -c $*.c

farm: $(OBJECTS)
[TAB] $(CC) -o farm $(OBJECTS)

cow.o: $(INCLUDES)

pig.o: $(INCLUDES)

farm.o: stdio.h
[TAB] $(CC) -c farm.c
```

In the example above, the lines

```
.c.o:
[TAB] xlc -c $*.c
```

says, to make a .o file from a .c execute the line

```
xlc -c $*.c
```

$* is a predefined macro that evaluates to the root portion of the file being made (the one to the left of the colon on the dependency line).

That is, given the dependency line

```
aaa.o : aaa.c b.h
```

$* evaluates to the string aaa. In the case of this generic dependency, make assumes that all .o files depend on a .c file having the same root name. That is, when aaa.o is being made, a dependency on aaa.c is assumed and need not be listed on the dependency line.

Some command line options for make:

-t "touch", causes all the last-modified times to be changed as if program had been recompiled. Useful when only comments have been changed.
-i ignore errors returned from subprograms. Normally an error would cause the compilation to terminate. If you use the -i option ensure that you redirect make's output to an error file.

The make utility

In its simplest form, make looks at 'dependency lines' in a file named 'makefile' in the working directory. The dependency lines indicate relationships between files, specifying a 'target file' that depends on one or more prerequisite files. If any of the prerequisite files have been modified more recently that its target file, make updates the target file base on 'construction commands' that follow the dependency line. Make normally stops if it encounters an error during the construction process.

Format of simple makefile:

target: prerequisite list [TAB] construction-commands

Example:

```
form: [TAB]     a.o b.o [TAB]     cc -o form a.o
b.o

a.o : [TAB]     a.c z.h [TAB]     cc -c a.c

b.o : [TAB]     b.c z.h [TAB]     cc -c b.c

z.h : [TAB]     t.h  u.h  [TAB]     cat t.h u.h >
z.h
```

Implied dependencies

If you do not include a dependency line for an object file, make assumes that it depends on a compiler or assembler source code file. Thus, if a prerequisite for a target file is XXX.o, and there is no construction command following a dependency line for the XXX.o target, make looks for one of the following files in the working directory:- XXX.c, XXX.f, etc. (see /etc/make.cfg). If it finds an appropriate source file, make provides a default construction command line that calls the proper compiler or assembler to create the object file.

Example:

```
#
# example make file
#

compute:[TAB]     compute.o calc.o
[TAB]             cc -o compute compute.o calc.o

compute.o:[TAB]   compute.c compute.h
[TAB]             cc -c -0 compute.c

calc.o:[TAB]      calc.c
[TAB]             cc -c calc.c
```

If you run make and no changes have been made to the prerequisite files, make does not execute any of the commands but it does print 'xxxxxx is up to date'.

Macros

Format of macro:- NAME=list
To use the macro in the makefile:- $(NAME)

Example:

```
#
# makefile: report, print, printf, printh
#

CFLAGS = -0
FILES = in.c out.c ratio.c process.c tally.c
OBJECTS= in.o out.o ratio.o process.o tally.o
HEADERS= names.h companies.h conventions.h

report:[TAB]     $(OBJECTS)
[TAB]            cc -c report $(OBJECTS)

ratio.o:[TAB] $(HEADERS)

process.o:[TAB] $(HEADERS)

tally.o:[TAB] $(HEADERS)

print: [TAB] pr $(FILES) $(HEADERS) | lp

printf: [TAB] pr $(FILES) | lp

printh: [TAB] pr $(HEADERS) | lp
```

The first dependency line shows that 'report' depends on the list of files that OBJECT defines. The corresponding construction line links the OBJECTS and creates an executable file named 'report'.

The next three dependency lines show that three object files depend on the list of files that HEADERS defines. There is no construction line, so when it is necessary, make looks for a source code file corresponding to each of the object files and compiles it. These three dependency lines ensure that the object files are recompiled if any of the header files is changed.

These three dependency lines lines could have been combined into one line:-

```
ratio.o process.o tally.o:[TAB] $(HEADERS)
```

The last three dependency lines send source and header files to the printer. They have nothing to do with compiling the 'report' file. None of the targets depends on anything. When you call one of these targets from the command line, make executes the construction line following it, example: make printf

SOME OTHER COMPONENTS

If an action is prefixed with a hyphen make will ignore errors in the action.

If the action is prefixed with '@' make is silent about the action, i.e. it is not echoed to the standard output.

Comments can be embedded in a makefile.

Other built in macros include:-

$* target name, minus suffix
$@ full target name
$< list of referred files
$? referred files newer than target

NATIONAL OS9 USER GROUP MEMBERS as at 09/07/93 mm/dd/yy

AMBROSI	G. A.	172 OGILVIE STREET	ESSENDON	VIC 3040
BAILEY	Eric	61 WINCHESTER STREET	MOONEE PONDS	VIC 3039
BARKER	Robert	P.O. BOX 711	LIVERPOOL	NSW 2170
BARTOLEC	Dubravko	13 McKAY STREET	DUNDAS VALLEY	NSW 2117
BENTZEN	Gordon	8 ODIN STREET	SUNNYBANK	QLD 4109
BERRIE	Don	"EAST END" MS 1445	MT LARCOM	QLD 4695
BLAZEJEWSKI	Stan	51 EVAN STREET	PARKDALE	VIC 3195
BOARDMAN	Margaret	U2/26 MONASH RD	PORT LINCOLN	SA 5606
CUNNINGHAM	Eric	7 NUTHATCH STREET	INALA	QLD 4077
DEVRIES	Bob	21 VIRGO STREET	INALA	QLD 4077
DONGES	Geoff	P.O.BOX 326	KIPPAX	ACT 2615
EATON	David	20 GREGSON PLACE	CURTIN	ACT 2605
GALL	Brian D	PO Box 131	COORANBONG	NSW 2265
GODFREY	Stephen	HEINEMANN ROAD	REDLAND BAY	QLD 4165
HOLDEN	Rod	53 HAIG ROAD	LOGANLEA	QLD 4131
JACQUET	J.P.	27 HAMPTON STREET.	DURACK	QLD 4077
KINZEL	Burghard	LEIPZIGER RING 22A	5042 ERFTSTADT	GERMANY
LARAWAY	Terry	41 N.W. DONCEE DRIVE	BREMERTON	U.S.A. WA98310
MACKAY	Rob	7 HARBURG DRIVE	BEENLEIGH FOREST	QLD 4207
McLINTOCK	George	7 LOGAN STREET	NARRABUNDAH	ACT 2604
McMASTER	Brad	P.O. BOX 1190	CROWS NEST	NSW 2065
MURPHY	Kevin	108 ADENEY AVE	KEW	VIC 3101
REMIN	Fred	11 CORCORAN CRES	CANUNGRA	QLD 4275
SINGER	Maurice	217 PRESTON ROAD	WYNNUM WEST	QLD 4178
SKEBE	Jeff	23 NORMA ROAD	PALM BEACH	NSW 2108
STEMAN	John	P.O.BOX 680	WINDSOR	NSW 2756
SWINSCOE	Robin	17 MELALEUCA STREET	SLADE POINT	QLD 4740
TUTELAERS	Peter	STRIJPERSTRAAT 50A	5595 GD LEENDE	NETHERLANDS
USERS GROUP, Inc	The OS-9	6158 W. 63rd Street	Suite 109 U.S.A.	CHICAGO ILL 60638
Total Members	29			